UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Implementación de la Visión artificial en cuadricópteros

Alumno: Guillermo Benéitez Ortega

Director: Omar Ait-Salem Duque

TÍTULO: Implementación de la Visión Artificial en cuadricópteros

AUTOR: Guillermo Benéitez Ortega

DIRECTOR DEL PROYECTO: Omar Ait-Salem Duque

FECHA: 14 de julio de 2023

INDICE

INDICE DE IMÁGENES

RESUMEN

El objetivo de este proyecto es programar una aplicación intuitiva para la automatización en la detección y geolocalización de objetos. Para lograr esta detección, se va a necesitar la utilización de un sistema de visión artificial para la que se requerirá una base de datos de imágenes. Este sistema va a contar con un cuadricóptero, una cámara, un software y hardware para la toma de fotografías sobre el objeto seleccionado. Una vez se han conseguido las imágenes se va a realizar el etiquetado de estas a través de una aplicación del propio software. Este recuadro va a generar una variable con la que se van a entrenar diferentes redes neuronales con el fin de conseguir la detección del objeto. Al conseguir el detector, y poder visualizar el objeto recuadrado, resta la parte de investigación de cómo realizar la geolocalización del objeto detectado. Para ello se van a plantear varias hipótesis para conocer la ubicación en coordenadas del objeto.

Palabras clave: *"visión artificial", "cuadricópteros", "redes neuronales", "detección" y "geolocalización"*

ABSTRACT

The goal of this project is to program an intuitive application for the automation of object detection and geolocation. To achieve it, an artificial vision system is used. This system is composed of a quadcopter, a camera, a software, and a hardware. In the first step, it takes images of the selected item by using the camara. Once the images are obtained, the system labels the photographs using an app installed of the software. This boxed object will generate a variable to be used in the training of different neuronal networks to obtain the object detection. Once it gets the detector, and the boxed object can be visualized, the research part of how to perform the geolocation of the detected object remains. To this end, several hypotheses will be considered to determine the ubication in coordinates of the object.

Key words: *"artificial vision", "quadcopter", "neuronal networks", "detection", "geolocation".*

1. INTRODUCCION

En este apartado se va a introducir el proyecto, explicando en líneas generales cuales son los objetivos para alcanzar. En este punto también se van a declarar, cuáles son los beneficios de utilizar el entrenamiento de redes neuronales para la detección de objetos en el ámbito de los cuadricópteros.

1.1 Objetivos

El objetivo de este proyecto es diseñar e implementar un sistema de redes neuronales, a través de una base de datos de imágenes para la detección de objetos y su posterior geolocalización. El modelo a construir detectará el objeto, marcándolo con un recuadro en tiempo real, para que el drone a través de una cámara consiga detectarlo y más adelante localice su posición exacta en coordenadas.

En este proyecto los objetos que se han planteado detectar son botellas de plástico, de cualquier tamaño y marca, empezando con las botellas de mayor tamaño por su facilidad a la hora del reconocimiento, ya que es más sencilla la detección de objetos de gran tamaño.

1.2 Beneficios del proyecto

El proyecto que se ha planteado presenta una visión diferente a los métodos por los que se ha realizado este tipo de trabajos. Actualmente, se utilizan diferentes softwares y métodos para poder realizar este tipo de tareas. La detección de objetos y el entrenamiento de redes neuronales están en crecimiento, lo que conduce a que la gran mayoría de las empresas de la industria 4.0 quieren que se introduzca esta tecnología.

Este proyecto está enmarcado en la labor de la empresa IBTICAE SL en el ámbito medioambiental de sostenibilidad mediante la recogida de residuos en campo abierto a través de drones e inteligencia artificial. Esto se consigue utilizando un cuadricóptero el cual va a realizar dos pasadas. En la primera pasada del dron por la zona, va a ir grabando con una cámara en tiempo real para detectar los objetos, en este caso botellas, que estén en el suelo en cualquier superficie y ubicar su posición en coordenadas. Una vez se tiene esta información, se hace una segunda pasada, en la que el dron de manera automática va a la zona donde están los residuos a retirar y lo recoge.

Este proyecto va a facilitar el aprendizaje de las redes neuronales utilizando un software libre a disposición de cualquier empresa o investigación. Para conseguir esto, se plantea un manual de uso de dicho software con todas las explicaciones necesarias para poder realizar el entrenamiento de redes neuronales y como utilizar la detección de imágenes [1][2].

Se quiere utilizar drones equipados con redes neuronales para la automatización y mejora de los distintos sectores que existen actualmente, desde la agricultura hasta el entretenimiento. Esto se debe a que los drones tienen mayor eficiencia, una máquina no comete fallos, por lo que un dron que aprende por sí solo, una máquina inteligente va a disminuir los errores que pueden producir los humanos. Si se sigue este hilo, los costes a corto plazo son mayores, ya que la inversión en esta tecnología es alta, pero a largo plazo, solo es necesario hacer el mantenimiento de los drones y de su código, por lo que se declara una opción más rentable a los usos ordinarios [1][2].

En este proyecto, se va a detectar un objeto y geolocalizarlo para conocer sus coordenadas a través de diferentes métodos de conversión de datos. Se hará uso de programación interna en las redes neuronales, para que se consiga que estos cálculos duren segundos, demostrando que utilizar dicha tecnología aumenta la velocidad a la hora de que se realicen tareas de recogida de objetos o envío a un punto concreto [3].

Uno de los beneficios más relevantes en el uso de la visión artificial en los drones es la eficiencia en los distintos campos de trabajo, sobre todo en tareas relativas al mapeo de zonas, ya que un dron que contenga esta tecnología solo necesitará una pasada para poder registrar toda la información relevante, y con una segunda pasada poder realizar cualquier operación de manera automatizada. Las tareas de recopilación de datos de superficies o de inspección de terrenos, son tareas que se tardan días y semanas, mientras que, con esta tecnología, se tardan horas en conseguir toda la información necesaria para poder actual sobre la misma [1].

2. Estado del arte

En este apartado se va a explicar cuál es el problema que se plantea a la hora de la realización de este proyecto, y como se quiere conseguir solventar utilizando un dron con visión artificial integrada. Para que se pueda conocer más acerca de esta tecnología se va a comentar su historia y como ha ido evolucionando hasta llegar a los drones que se conocen hoy en día.

2.1 Problemática

A la hora de iniciar este proyecto, se precisó tener una reunión con la empresa con la que se va a trabajar este proyecto, IBTICAE. En dicha reunión se comunicó como se había planteado este proyecto, se quería detectar un objeto concreto y más adelante su localización en el plano de coordenadas. Se comentó que este mismo proyecto ya se había realizado con unas herramientas más cerradas, y queriendo comprobar cómo se comportan herramientas más abiertas para la realización de este proyecto.

Una vez se ha analizado el trabajo que presenta la empresa, se han extraído varios puntos que se tienen que solucionar para poder completar la tarea de manera satisfactoria. Las incidencias encontradas a la hora de examinar el proyecto son las siguientes:

- Este tipo de proyectos se han realizado utilizando herramientas más cerradas como por ejemplo la inferencia de Yolo o se ha utilizado Google Collab, por lo que se desconoce cómo va a ser el comportamiento en los softwares más abiertos a la hora de obtener los resultados.

- Uno de los problemas más relevantes, es que el espacio de trabajo no es uno fijo, ya que el drone se va a ir moviendo por distintas zonas y va a complicar el entrenamiento de las redes ya que, al cambiar el fondo y el entorno en todo momento, se va a tener que debatir si se utilizan métodos de filtrado de imágenes debido a la cantidad de información con la que cuenta la naturaleza y los errores que se pueden llegar a encontrar. Por otra parte, se requiere detectar todos los tipos de botellas, por lo general se empezará con las de plástico. Por lo que se va a necesitar una cantidad muy grande de imágenes y tener un ordenador lo suficientemente potente para entrenar tanta información en diferentes ubicaciones.

- Otro conflicto que genera este proyecto es la creación de cero de una interfaz que sea sencilla de utilizar, y que cuente con un manual para su uso. Este punto no se ha realizado anteriormente en un software libre, la creación de un procedimiento explicando paso por paso lo que se va realizando en todo momento, dejando un código limpio y cómodo para el consumidor.

- El último problema que se ha abordado es la geolocalización de objetos en tiempo real. Este punto presenta varias dificultades para compatibilizar los algoritmos de inteligencia artificial con el hardware enmarcado en el dron.

Una vez se ha analizado la problemática planteada por parte de la empresa, se delibera que software utilizar para cumplir con los requisitos que solicita la empresa IBTICAE. Por ello se eligió MATLAB, ya que al ser una herramienta más abierta y que se ha utilizado durante el curso académico, se conoce su funcionamiento y los métodos tanto de entrenamiento de imágenes a través de redes neuronales como los métodos de filtrado de las fotos para poder llevar a cabo el programa solicitado.

2.2 Historia/Antecedentes

El comienzo de los drones llegó antes de lo que se piensa, tema que se tratará durante este punto, pero antes de empezar la historia de estos vehículos se necesita saber que un dron es un pequeño vehículo el cual no cuenta con una tripulación, se diseñó para el campo militar, pero se está desarrollando para temas civiles, sobre todo rescate [4].

Se puede pensar que los drones son algo actual, pero no lejos de la realidad, se dieron en 1843 donde se empezó a analizar y diseñar una especie de avión, pero este prototipo no contaba con piloto, por lo que el inventor William Henson tuvo que desestimar lo construido, el primer vehículo aéreo que contaba con hélices, motores e incluso tenía un ala y la característica de no utilizar un piloto, pero al hacer pruebas y corroborar que solo duraban pocos segundos en el aire, por lo que se desechó tanto el prototipo como esta idea [4].

Los primeros drones construidos y utilizados se dieron como una fuerza militar innovadora, datan de 1849 cuando Austria uso globos no tripulados con explosivos para atacar Venecia. Estos globos, se encendían a través del electromagnetismo, gracias a un cable de cobre unido a una batería galvánica, la única instrucción que se tenía era que se lanzasen los globos cuando los vientos fuesen favorables en la dirección del objetivo. Pero un viento repentino hizo que todos los globos se desviaran lejos del verdadero objetivo [5][6][7].

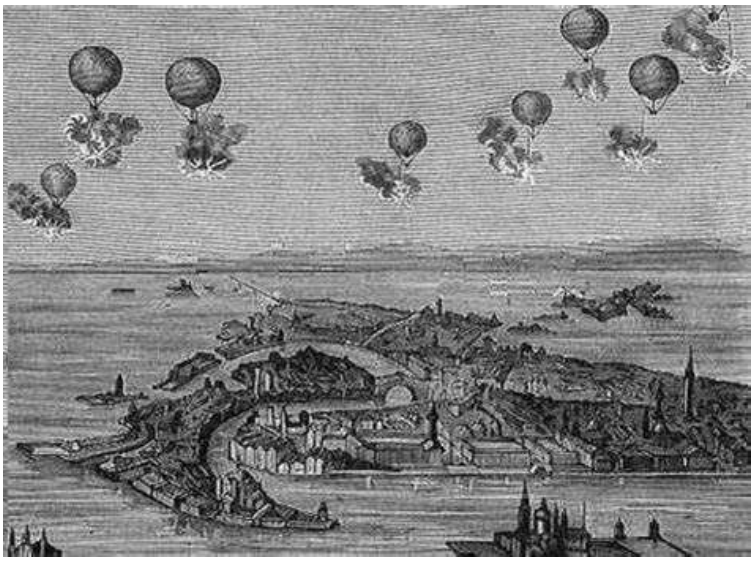

Ilustración 1 – Primeros globos no pilotados

Como se ha mencionado anteriormente, los drones se usaban única y exclusivamente en el ámbito militar por lo que su innovación y evolución era secreta para no dar esa información al enemigo. Por ello no se tuvo ninguna señal nueva de esta tecnología hasta 1858, con la revolución de las telecomunicaciones, esta revolución vino dada por la aparición del telégrafo, esto afecto en la en las ondas electromagnéticas que se utilizaban para encender los globos que se han comentado anteriormente. Se descubrió la posibilidad de comunicación a través de estos campos mediante la frecuencia o amplitud de las ondas de radio [5][6][7].

Durante el resto del siglo XIX, no se dieron avances relevantes en el ámbito de los drones ni de los aviones, por lo que la entrada del siglo XX daría el auge de los vehículos aéreos, donde más avances hubo sobre todo en el entorno militar debido a las dos guerras mundiales, en las que la innovación lo era todo [5][6][7].

En la actualidad, lo más utilizado y moderno en drones son los cuadricópteros (drones que constan con 4 rotores externos) y no fue hasta 1907 donde se diseñó y se construyó el primer cuadricóptero por parte de los hermanos Jacques y Louis Bréget. Este avance tecnológico dio cabida al helicóptero, pero este vehículo estaba muy lejos de la tecnología con la que se contaba en la época, por lo que se hicieron pruebas con el primer drone, consiguiendo resultados en altura de 600 centímetros en un vuelo vertical sin contar con piloto, pero se necesitaron cuatro personas para conseguir la estabilización del vehículo. Con este primer drone, se consiguió como conclusión la posibilidad de poder utilizar esta tecnología lo único que se necesitaba para lograrlo es un pequeño avance tecnológico [5][6][7].

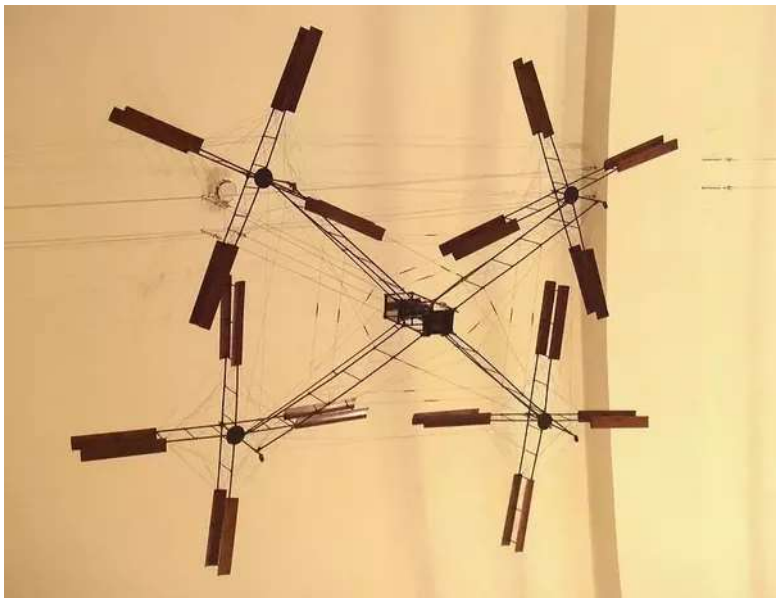

Ilustración 2 - Primer cuadricóptero construido

La importancia de esta tecnología se pudo dar con la entrada de la primera guerra mundial (1914-1918) ya que la innovación tecnológica en este ámbito fue muy significativa, desarrollando así el primer vehículo aéreo no tripulado en 1916, solucionando los problemas del despegue vertical del drone el cual se construyó por los hermanos Bréget con un autopropulsado, pero en este caso, utilizando aire comprimido para conseguir este despegue, con las siguientes pruebas, el drone acabó estrellándose debido a problemas en el motor, pero dejando la posibilidad de seguir avanzando en este tipo de tecnología. Este nuevo vehículo, fue construido por un equipo elegido de aproximadamente 30 personas y le pusieron el nombre de Ruston Proctor Aerial Target. Como se ha indicado anteriormente, todos los avances en el tema de los drones se utilizaban en la guerra, por ejemplo, el sistema diseñado de despegue vertical con aire comprimido se utilizó para el despegue de los cohetes, y luego ese cohete se dirigía mediante la radio para llegar a su objetivo [5][6][7].

Durante el 1916 y 1917, el ejército de Estados Unidos, también aposto por la tecnología de no usar piloto, pero esta vez, se decidió utilizar un control giroscópico en un dispositivo capaz de lanzar torpedos. Capaces de controlar de manera muy precisa el lanzamiento de parábola y cuando se llegaba al punto requerido, dichos torpedos se daban la vuelta, dejando de avanzar y cayendo en picado, una tecnología de control a distancia necesaria a la hora del control de vehículos. Esto se consiguió gracias a Elmer Ambrose Sperry el desarrollador del giroscopio (creado en 1852 por Foucault), lo que conocemos en la actualidad como el piloto automático de la época, usado para mantener o cambiar la orientación del vehículo o aparato o incluso para medir distancias [6][7].

Ilustración 3 - Giroscopio

Una vez finalizada la primera guerra mundial, se empezaron los avances significativos, ya que como se ha comentado anteriormente, lanzar o mover aparatos militares sin la necesidad de un piloto y de manera remota, impulso a algunas potencias a avanzar en este tema. La primera fue Estados Unidos, que desarrollo sin cesar este tipo de drones hasta que en 1930 empezaron los experimentos sobre las comunicaciones con el drone mediante radio consiguiendo tras unos años de experimentos construir el dron Curtis N2C-2 en 1937. Otra de las potencias que no dejaron escapar los avances de los drones fue Inglaterra, desarrollando el Queen Bee, un dron dirigido por radio, el primer dron conocido como UAV capaz de hacer dicha función, utilizado como objetivos aéreos en las misiones de entrenamiento. Con este dron, se tuvieron los mejores resultados, siendo una referencia para los futuros avances ya que consiguió llegar hasta una altura vertical de más de 500 metros y un avance de 480 kilómetros. Debido a estos resultados se construyeron alrededor de 380 Queen Bees, sirviendo para todas las misiones aéreas hasta el 1947, que retiraron este tipo de drones [5][6][7].

Entrando en la segunda guerra mundial (1939-1945) volvió a aparecer la figura de los Estados Unidos, en esta ocasión, juntaron la tecnología de los UAV con el reconocimiento, ya que tenían como misión reconocer los bunkers alemanes a través de bombardeos readaptados, los cuales estaban dirigidos por control remoto para poder explotar en contra de dichos bunkers. Este tipo de bombarderos seguían teniendo los problemas antiguos, solo se les podía sacar provecho cuando no hay ningún problema atmosférico ya que con el control remoto se encontraban muchas interferencias y se complicaba el control, debido a eso, en ocasiones los paracaidistas se tenían que hacer cargo de dirigir el vehículo. Este movimiento de los bombarderos, se denominó Operación Anvi, el cual se desechó ya que cuando se llevó a la práctica en el territorio enemigo, muchas de las aeronaves se estrellaron por lo que la tecnología seguía siendo limitada para entonces, pero con mucho potencial para las próximas décadas [6][7].

Los siguientes usos de los drones no tripulados, se dieron con el espionaje, tras el caso Gary Powers ya que entre las décadas de 1950 y 1960, pudiendo espiar donde la presencia de un piloto estadounidense tenía sus desventajas, ya que al mínimo contacto en el radar de la Unión soviética daba lugar a ser un vehículo derribado en una tarea donde no era necesaria tanta precisión. Debido al suceso que se acaba de comentar, el presidente de los Estados Unidos decidió desarrollar las naves no tripuladas. Se empezó con el Gran Safari el primer drone de vigilancia construido por la empresa Ryan Aeronautic, se siguió utilizando el despegue vertical a presión como se usaba con el aire comprimido, pero eso no se convertía en una prioridad, porque para las pruebas se lanzaban los drones desde un avión para probar las rutas que se tenían programadas y no solo eso, también se podían controlar remotamente pero solo por los integrantes del avión desde donde se lanzaban dichos UAVs. Para terminar su prueba, se programaron los drones para desplegar un paracaídas para que no se pierda la tecnología y que se pueda recoger con el fin de su seguimiento con las pruebas [6][7].

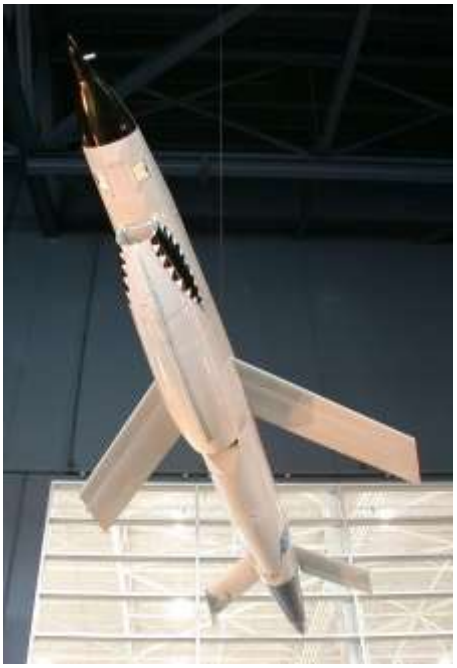

Ilustración 4 - Drone de espionaje

Estos drones, se utilizaron en las misiones durante uno de los conflictos más grandes y largos, la Guerra Fría, por ello se requería de tecnología de vigilancia más complejos. La Guerra Fría se proclamó como la primera guerra tecnológica de la historia, tanto por el tema de los UAVS como de los sistemas electrónicos. Esto supuso que Estados Unidos durante 1960 comience la automatización de los sistemas electrónicos. Uno de los ejemplos de esta iniciativa, fue usar sensores y superordenadores con el fin de conseguir información tanto por sonidos como la ubicación de los enemigos y, además, poder manejar desde control remoto los drones no tripulados.

La foto que se observa es el Firebee, el UAV más se ha usado por los estadounidenses, llegando a los 30 años de uso y se cumplieron más de 3500 misiones de espionaje entre 1964 y 1975. Estas misiones se quedaron dentro de la operación llamada Igloo White, en la cual para conseguir todo el avance tecnológico se aportó una inversión de millones de euros. Esa aportación económica dio lugar a que, a partir de 1972, los Firebee contaran con un equipamiento LORAN (Long range navegation), un sistema de navegación mejorado debido a un sistema electrónico de ayuda el cual consta de un receptor, capaz de interpretar y utilizar las señales de tres o más transmisores para establecer la ubicación aproximada del receptor [6][7].

Estados Unidos seguía acaparando todos los avances tecnológicos de los drones, esto se debe a la gran financiación que aportaba, en este caso a las compañías Boeing y Ryan. La idea en esta ocasión era crear drones de vigilancia más resistentes y con mayores capacidades de altura y vigilancia, esto supuso el desarrollo de UAVs capaces de aguantar 24 horas en el aire, debido a su menor peso, objetivo que se consiguió al reducir de tamaño y se le pudo incorporar sensores y una cámara de video para poder extraer más información [6][7].

Como se ha comentado anteriormente, los grandes avances en el desarrollo de drones vienen dados por Estados Unidos, consiguieron poder crear en masa UAVs para potenciar su poder militar hasta que se llegó a un punto en el que los drones se llegaban a considerar demasiado costosos y además con carencias en su fiabilidad a la hora de obtener resultados. Está perspectiva se cambió en 1982 cuando las fuerzas militares sirias, decidieron comprar los drones no tripulados a Estados Unidos para librar una guerra aérea en el bando comprador salió victorioso con pocas pérdidas. Al ver los resultados, Estados Unidos comenzó a desarrollar drones más económicos para las operaciones de flota iniciando así el programa Pioneer UAV en 1980, con el fin de obtener informaciones espaciales [6][7].

Estados Unidos e Israel cooperaron en un proyecto conjunto en 1986 condujo al desarrollo del RQ2 Pioneer, un UAV de reconocimiento de tamaño mediano. Con los avances en el desarrollo del proyecto debido a la evolución de los microchips, se conseguía un sistema electrónico superior, pero en menos espacio y menor peso, los fabricantes empezaron a buscar fuentes de energía distintas para drones, por lo que, la idea más relevadora ya que son UAVs para el espacio, el resultado fue energía solar [6][7].

Ilustración 5 - UAV de reconocimiento

En la misma década, fue cuando los drones se comenzaron a desarrollar y usar en el ámbito civil, comenzando en la industria y en el cine. Los drones ya contaban con una cámara equipada, la problemática del uso civil se basaba en la duración de las baterías y en la calidad de la cámara, ya que, al usar drones más pequeños, solo podían cargar con cámaras con poca calidad. El primer pensamiento a la hora de equipar a los drones con una cámara no era con un fin cinematográfico sino con un fin de espía como ha comentado anteriormente. A partir de 1990, fue cuando se desarrollaron los sistemas en los drones para ser capaces de llevar una cámara con mayor calidad para su uso. El problema de la batería no se tardaría en solucionar, ya que, en la misma década, así fue como la tecnología de consumo se involucró en el desarrollo de los drones utilizando unas mejores baterías, las cuales se siguen usando hoy en día, las baterías de Litio. En las primeras pruebas, comprobaron que solo se necesitaban cargar 5 minutos la batería para lograr un vuelo de al menos 30 minutos, este suceso logro abrir muchos ámbitos para el uso de los UAVs [6][7].

Durante todos los años y etapas que han transcurrido la vida de los drones, se han utilizado con objetivos militares e inicios de objetivos recreativos. Al inicio de la década del 2010, se visualizó un nuevo uso para los drones, entre ellos el utilizarlos como mensajeros de objetos o cartas, por lo que el interés comercial de los mismos aumentó [6][7].

Para utilizar los drones de manera personal o de otra manera, se necesita una licencia, lo que llevo a acarrear un aumento de las licencias, ya que en 2015 se emitieron por parte de la Administración Federal de Aviación (FAA) alrededor de 1000 licencias comerciales. Tres años más tarde, se volvió a dar la situación de las licencias debido a comercializar los drones con cámaras incluidas, para poder hacer fotografías y videos, consiguiendo así la mezcla de los UAVs y la visión artificial [6][7].

Ilustración 6 - Drones comerciales con cámaras

En la actualidad, el ámbito en el que los drones participan se ha aumentado exponencialmente debido a que las industrias se han dado cuenta de la utilidad con la que cuenta esta tecnología. En 2019 Amazon presento su propia y mejorada versión de un dron en la conferencia MARS (Machine Learning, Automation, Robotics and Space), el Prime AIR. Este dron, está diseñado para revolucionar el mundo de la mensajería ya que lo que se propone por parte de Amazon es que realice el mismo las entregas, reconociendo el pedido dentro del almacén y llevando al destino de manera autónoma. Esto es posible ya que lleva implemnetadas varias cámaras y varios sensores que le ayudan a evitar obstáculos gracias a las técnicas del entrenamiento de redes neuronales de Machine Learning y Deep Learning y, además, conocer su ubicación actual en cada momento para poder llegar al destino [5][16].

Ilustración 7 - Prime Air

3. Sistema de obtención de imágenes

Para poder llevar el proyecto a cabo, es necesario seguir unos pasos antes, desde la obtención de imágenes hasta el uso de estas dentro de las redes neuronales con el objetivo de poder localizar los objetos, para ellos se ha necesitado utilizar un sistema de obtención de imágenes. Para lograrlo, se va a utilizar un sistema constituido por un hardware y un software que se explicarán en los siguientes apartados.

3.1 Hardware

Una de las partes más relevantes de este proyecto es la obtención de imágenes para poder crear un detector propio entrenando dichas fotografías y alimentando una red neuronal. Para lograrlo es necesario contar con imágenes tomadas anteriormente y poder tomar imágenes en tiempo real por lo que se va a necesitar contar con los siguientes elementos físicos.

La toma de imágenes se hizo a través de un dron, en este proyecto, se ha trabajado con la IBTICAE, una empresa la cual aportó un cuadricóptero de elaboración propia para poder obtener las imágenes. Las características del dron utilizado son las siguientes:

- Model: Flame Wheel 550 (F550)
- Frame Weight: 478g
- Diagonal Wheelbase: 550mm
- Takeoff Weight: 2400g
- Battery: 4S LiPo • System:
- Battery (Recommended): 4S LiPo
- Max. Thrust: 850 g/rotor (14.8 V, Sea Level)
- Takeoff Weight (Recommended): 350 ~ 400 g/rotor (Sea Level)
- Working Temperature: -10 ~ 40°C
- Max. Allowable Voltage: 17.4 V
- Max. Allowable Current (Persistent): 20 A
- Max. Allowable Peak Current (3 seconds): 30 A
- PWM Input Signal Level: 3.3 V /

Este cuadricóptero no solo consta de las características mencionadas anteriormente, si no que, tiene incluido una Jetson Nano y una cámara. La Jetson Nano es una minicomputadora, la cual permite que se pueda correr programas en su interior, especialmente, está diseñado para el entrenamiento de redes neuronales sobre todo para la detección de objetos. Por ello, se va a incluir en el dron, para no tener un ordenador conectado al cuadricóptero ya que el cable afectaría al recorrido de este [11][12].

Como se acaba de mencionar, lo último que está incluido en el dron es una cámara, el elemento más importante, ya que, gracias a ella, se va a poder realizar la toma de imágenes y grabar en todo momento lo que ocurre para poder tener resultados en tiempo real. Para lograr esto, la cámara tiene que contar con una alta resolución de video de hasta 8 MP (megapíxeles) y con un zoom variable de entre 5 y 50 mm, lo cual se va a utilizar cuando se gane altura, regulando tanto el zoom como la resolución. Las características de la cámara elegida son las siguientes [15]:

Modelo: ELP-USB8MP02G-SFV(5-50)			
Resolución y marco	3264X2448 a 15 fps / 2592X1944 a 20 fps 2048X1536 a 20 fps / 1600X1200 a 20 fps 1280X960 a 20 fps / 1024X768 a 30 fps 800X600 a 30 fps / 640X480 a 30 fps		
Sensor	SONY IMX179	Formato de compresión	MJPEG/YUV2(YUYV)
Tamaño de la lente	1/3,2 pulgadas	Parámetro de la lente	Lente varifocal de 5-50 mm
Tamaño de píxel	1.4um	Tamaño/Peso de la placa	41*41mm, alrededor de 0,3 kg
área de imagen	6,18 mm x 5,85 mm	Tipo de puerto de conexión	USB 2.0 de alta velocidad
máx. Resolución	3264 (H) X 2448 (V) 8 megapíxeles	Protocolo de conducción libre	Clase de vídeo USB (UVC)
Relación señal/ruido	34dB	protocolo OTG	USB 2.0 OTG
Gama dinámica	72.5dB	Tipo de obturador	Persiana / exposición del cuadro
Sensibilidad	0.65V/lux-seg@550nm	AEC,AEB,AGC	Apoyo
Mini iluminación	0,5 lux	Cámara de botón	Apoyo
Tensión de funcionamiento	CC5V	Temperatura de trabajo	-10~70 grados
corriente de trabajo	150mA~240mA	Temperatura de almacenamiento	-20~85 grados
Ajustable parámetro	Brillo, Contraste, Saturación, Nitidez, Gamma, Ganancia, balance de blancos, tono, Contraste de luz de fondo, Exposición	Interfaz de control de placa LED	Admite enchufe 2P-1,25mm
		Conector de alimentación de la placa LED	Soporta enchufe 2P-2.0mm
Sistema operativo compatible	WinXP/Vista/WIN7/WIN8 / Linux con UVC Mac-OS X 10.4.8 o posterior Wince con UVC, Android 4.0 o superior		

Ilustración 8 - Características de la cámara

Una vez escogido y explicado las funciones de la cámara y el cuadricóptero, se va a comentar donde se van a realizar todos los cálculos y entrenamiento de las redes neuronales. En este caso se va a utilizar un ordenador sobremesa el cual cuenta con una tarjeta gráfica NVIDIA Geforce GTX 1650, con una velocidad de almacenamiento de datos de 8 GB (Gigabyte), un procesador CPU AMD Ryzen 5 2600 y con una memoria RAM de 16 GB dividida en dos memorias de 8 GB cada una. Es necesario usar un ordenador potente (el cual conste de una tarjeta gráfica potente) ya que el entrenamiento de redes es un proceso muy lento el cual se puede agilizar si se cuenta con un ordenador con GPU, para ahorrar tiempo en la velocidad de entrenamiento.

Como el entrenamiento de redes supone una programación y una necesidad alta de recursos físicos, va a provocar un calentamiento extra de los objetos utilizados en el ordenador anteriormente, lo que supondría que dichos componentes, se irán desgastando con el paso del tiempo e incluso con un único entrenamiento de gran capacidad. Por lo que es necesario tener uno o varios disipadores potentes con el fin de no sobrecalentar el ordenador y poder realizar el aprendizaje de las redes sin comprometer el equipo físico.

3.2 Software

Anteriormente se han mencionado varios métodos y distintos softwares por los que se ha decidido realizar este proyecto con el software Matlab, una de las plataformas de MathWorks diseñadas para la programación y los cálculos matemáticos. En este proyecto tendrá un enfoque de programación más enfocado al entrenamiento de redes neuronales gracias a las aplicaciones que incorpora este software y que se explicará el proceso de entrenamiento en el siguiente apartado [32].

Ilustración 9 - Matlab

4. Procedimiento

Antes de comenzar con la explicación del procedimiento, el cual incluye el etiquetado y entrenamiento de las redes, es necesario tener instalado el software de Matlab, a poder ser con la última versión, para contar con las últimas actualizaciones que pueda contar. Cuando se esté instalando Matlab, se preguntará sobre las toolbox necesarias para el entrenamiento, por lo que las requeridas son:

- Deep Learning toolbox
- Computer Vision toolbox -> apps etiquetado
- Image processing toolbox
- Statistic and machine learning toolbox
- GPU Coder

4.1 Procedimiento del Etiquetado

El primer paso para comenzar el etiquetado es adquirir muchas imágenes para que la red sea más estable, en este caso, se han utilizado alrededor de 1400 imágenes, es muy importante que las imágenes tengan las mismas dimensiones, ya que hay redes que no entrenan fotos de diferentes tamaños, en este ejemplo todas las imágenes tienen un dimensionado de 1024 x 1024. Una vez con el Software instalado, se inicia el programa y se busca en la pantalla principal en la zona superior de las herramientas que contiene Matlab la pestaña de APPS.

Ilustración 10 - Herramientas Matlab

Para realizar el etiquetado hay dos opciones, utilizar Image Labeler o Video Labeler, en este caso, se ha utilizado Video Labeler ya que con la otra herramienta los resultados son menos fiables. Una vez elegida, cargamos las imágenes pulsando en la opción Import, para cargar todas las imágenes se va a dar la posibilidad de elegir entre incluir las fotos, como un video o como una secuencia de imágenes. Por lo que elegiremos la segunda opción.

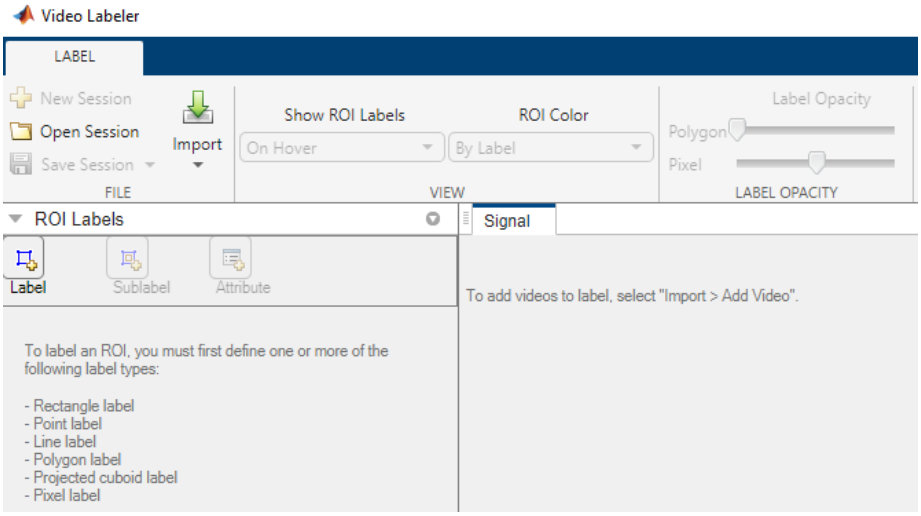

Ilustración 11 - Video Labeler

Para comenzar el etiquetado, hay que definir la variable que se va a elegir, en este caso, al ser un reconocimiento de botellas de plástico, la etiqueta que se va a utilizar es bottle. Por ello, se hace clic sobre en el cuadrado Label, y se define tanto el nombre como el color que se quiere tener en la salida de las imágenes.

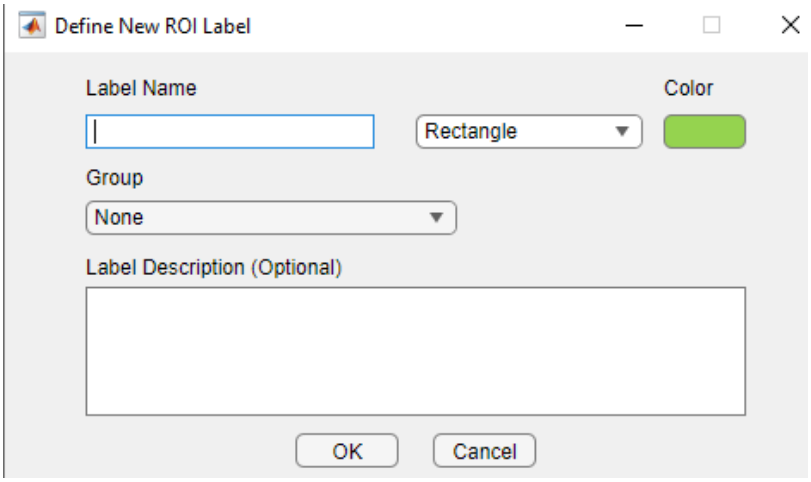

Ilustración 12 - Definir etiqueta

Una vez se ha definido la etiqueta, se va a comenzar a realizar el recuadro de cada una de las imágenes como veremos en la siguiente imagen, se hará el recuadro sobre cada uno de los objetos definidos en la etiqueta, se recomienda utilizar el rectángulo para etiquetar los objetos, ya que facilita la salida y el entrenamiento de las redes.

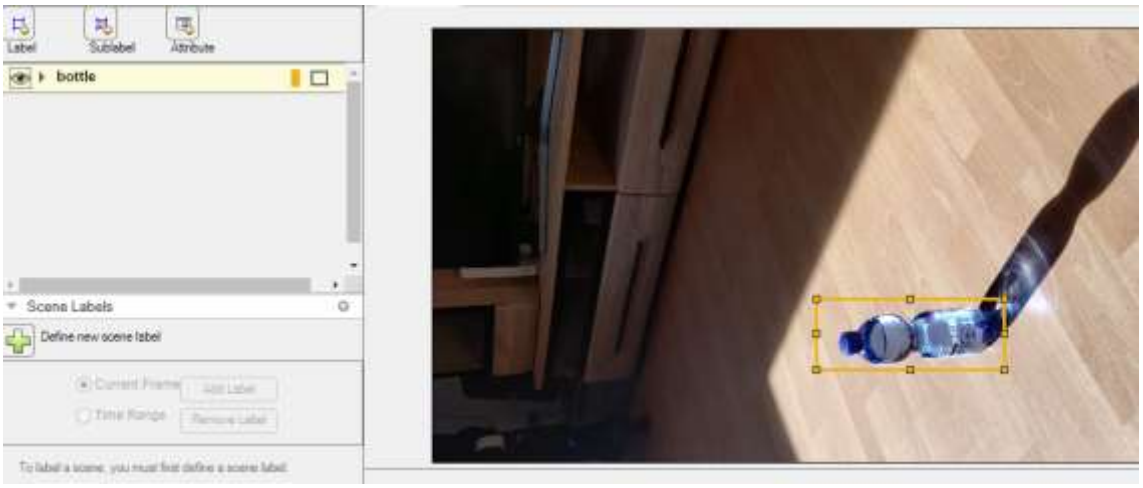

Ilustración 13 - Etiquetado para la detección

Este proceso de etiquetado se puede automatizar o realizarse de una manera más rápida si se usa el modo automático, para ello, en el panel de herramientas de Video Labeler, está la opción de automático, el cual nos mostrará opciones de algoritmo.

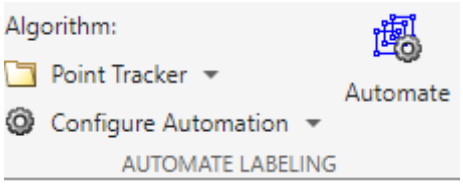

Ilustración 14 - Etiquetado automático 1

En este caso, se ha utilizado el Point Tracker, ya que se realiza el etiquetado desde unos puntos concretos, cuando se observa objetos parecidos, dependiendo del tipo de objetos o el tipo de imágenes que tengas, el resto de las configuraciones serán más o menos aptas.

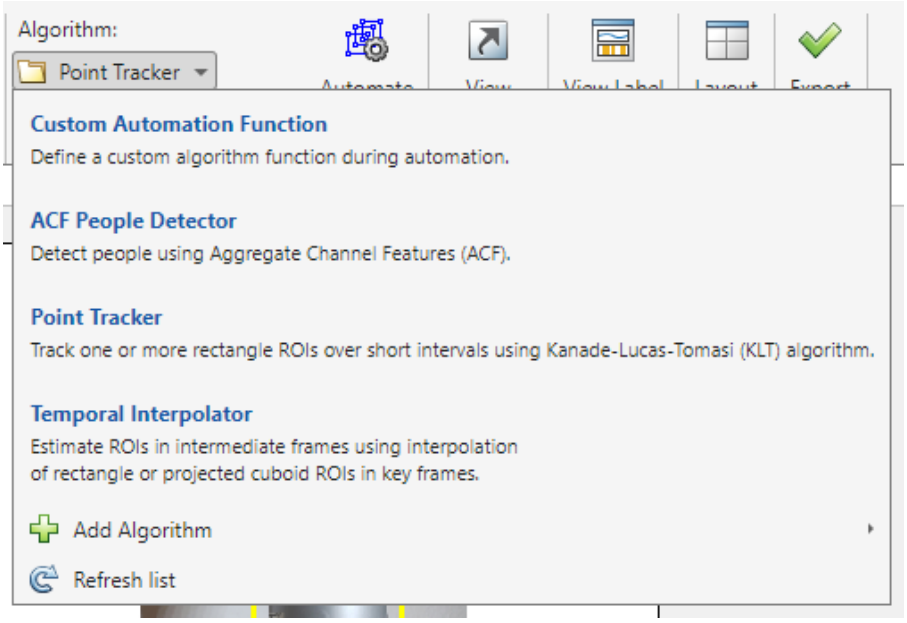

Ilustración 15 - Etiquetado automático 2

Una vez escogido el tipo de algoritmo, se tiene que elegir la configuración de automoción, si se va a realizar de inicio a fin o de fin a inicio, por una parte. Por la otra, de qué manera va a ir la automoción. Para este caso se recomienda la opción de Start time to End time, ya que se define desde donde hasta cuándo se va a realizar, así se agrupan las imágenes más parecidas o iguales entre sí.

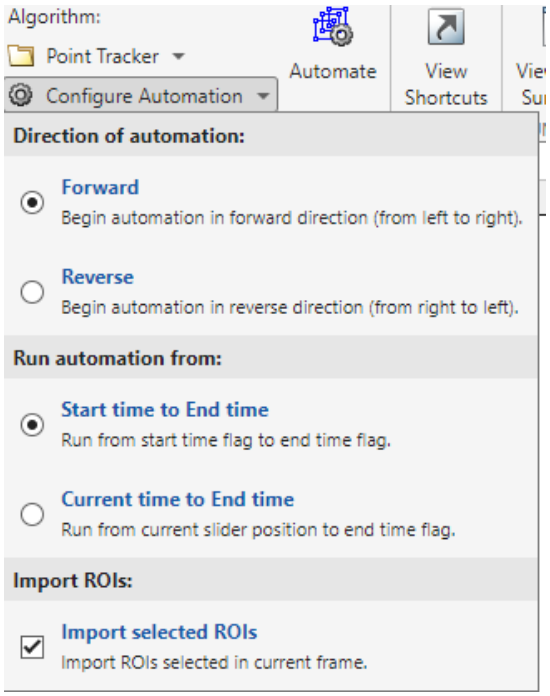

Ilustración 16 - Etiquetado automático 3

Cuando se ha realizado el etiquetado de todas las imágenes, se comprueba que todas las imágenes están bien recuadradas y los objetos a estudiar dentro de dichos recuadros. Al finalizar la comprobación, se exporta la sesión en la barra de herramientas clicando en Export Labels. Al pulsar en dicha opción, se abrirá un recuadro para poner el nombre a una variable, en este caso se utilizará gTruth como variable. Al exportar dicha variable al Workspace (donde se encuentran todas las variables de Matlab al correr un programa). Para guardar esa variable en nuestra carpeta y no tener que realizar este proceso en más ocasiones, se arrastrará el gTruth desde el Workspace hasta el Current Folder, y se cargará en el programa.

4.2 Procedimiento de la Programación

En esta ocasión, se han utilizado dos programas con redes neuronales diferentes para poder comparar distintos resultados y cuál de ellos otorgaba mejores resultados para su realización. El primer programa que se va a explicar es un entrenamiento de imágenes y como resultado se va a obtener el nombre del objeto que hemos etiquetado, en este caso saldrá un recuadro en el cual pondrá Bottle, mientras que, en el segundo programa, el resultado tras el entrenamiento se va a dar como el porcentaje de acierto del programa.

4.2.1 Programa recuadrado Red Simple

Como se ha comentado anteriormente, se ha creado un programa para el reconocimiento de botellas, para ello, en este caso, se ha dividido en tres partes, la zona entrenamiento, zona visualización y zona ubicación. Las cuales se explicarán a continuación.

El primer paso a seguir es la limpieza de variables, con los comandos clc y clear all como se observa en la imagen siguiente. A partir de la limpieza, se va a continuar en el punto que termina la parte del etiquetado, cuando cargamos el gTruth. Para ello, una vez guardado en la carpeta, se va a cargar con el comando load, donde se encuentra nuestro etiquetado el cual se definirá con la variable que se quiere localizar en el entrenamiento.

Depende de la red neuronal que se use para el entrenamiento se va a necesitar unas variables diferentes. En este caso, usando los datos del etiquetado, y usando los datos positivos del mismo, en este caso, hay varias opciones en la segunda parte del entrenamiento, pero a través de prueba y error, los mejores resultados que se han obtenido han sido con los resultados positivos, aunque para este tipo de proyectos, se suelen utilizar la estrategia de quitar los resultados del etiquetado negativos, este dato se obtiene dentro del entrenamiento del ACF detector.

Una vez se tienen los dos datos, se puede entrenar el detector, este proceso no va a tardar mucho, ya que al coger solo los datos positivos muchas de las imágenes que se han etiquetado se van a ver como negativos y no se van a utilizar.

```
1   clc
2   clear all
3
4   %entrenamiento faster
5   load('gTruth.mat')
6
7   %definir etiquetado
8   gTruth.LabelDefinitions
9
10  bottleGroundTruth = selectLabelsByName(gTruth,'bottle');
11
12  trainingData = objectDetectorTrainingData(bottleGroundTruth);
13  summary(trainingData)
14
15  %entrenar el detector
16  acfDetector = trainACFObjectDetector(trainingData, Verbose= true);
17
18  trainingDataTable = objectDetectorTrainingData(gTruth);
19  trainingData=trainingDataTable;
20
```

Ilustración 17 - Programación detector simple

Como resultado en esta parte se obtiene el detector, donde se registran todos los datos del etiquetado que hemos realizado en la app. Una vez tenemos el detector, se cambia a la zona de visualización, ya que, en Matlab, en la zona de herramientas se puede correr el programa por las zonas que se han creado pulsando run section, con el fin de no estar entrenando redes todo el rato.

En esta zona, se va a ver el comportamiento del detector, comprobando con imágenes tanto de dentro de la red como de fuera de la misma el desempeño de este. Para ello, se incluirá una imagen que se tiene externa del entrenamiento a través del comando imread. Si se quiere usar una foto tomada en el instante en vez de una imagen guardada en la carpeta, se definiría primero que cámara se va a utilizar, con el comando webcamlist. Luego se definiría una varia vinculada a la cámara que se ha seleccionado y como en este caso, se define la variable I como un snapshot de lo que registra la cámara.

Para terminar, el detector pasa por encima de la imagen para realizar el reconocimiento, anotando el objeto a detectar de la imagen de pruebas, y con el comando imshow se muestra el resultado, comprobando de esta manera si el detector entrenado es fiable.

```
24  %% zona visualizar
25
26  I = imread('Prueba7.JPG');
27  bboxes = detect(acfDetector,I);
28
29  annotation = acfDetector.ModelName;
30  I = insertObjectAnnotation(I,'rectangle',bboxes,annotation);
31
32  figure
33  imshow(I)
34
```

Ilustración 18 - Programación visualización

La última zona del programa es la zona de localización, ya que una vez se ha detectado el objeto, se geolocalice. Para ello se ha creado un apartado con una prueba de cómo se podría realizar. Al tener los datos de la altura a la que está el dron y la distancia a la que está el objeto (medidas con anterioridad al tomar las imágenes), se utiliza el teorema de Pitágoras para conseguir el dato de la hipotenusa ya que, uniendo la altura y la distancia al objeto, se forma un triángulo. Al resolver este teorema, se obtiene la distancia del objeto detectado y el dron.

```
37  %% zona ubicación
38  Vertical = 4;
39
40  Horizontal = 2,5;
41
42  Dist_drone = sqrt(Vertical^2+Horizontal^2)
43
```

Ilustración 19 - Programación localización

4.2.2 Programa recuadrado Red Compleja

En este programa, se va a utilizar la misma estructura que en el anterior, la zona de entrenamiento, zona de visualización y la zona de ubicación, de las cuales la secuencia del programa es la misma.

Como se ha realizado en el caso anterior, al terminar el etiquetado y tener la variable gTruth guardada donde queda recopilada la información. Antes de empezar con el entrenamiento se va a limpiar la zona de las variables para que no influya en los resultados que se van a obtener, una vez se ha realizado la acción se carga la variable mencionada en el programa con el comando load.

Para este entrenamiento se necesitan tres cosas, un detector el cual se va a conseguir a través de la variable gTruh, recopilando los etiquetados de este y el número de variables, la capa en la que se va a entrenar, la cual se pueden encontrar dentro de la propia red, en este caso, se ha utilizado la capa por defecto para ver el comportamiento y por último las opciones de entrenamiento.

```
 clc
 clear all

load('1024botellas.mat');
load("gTruth3.mat");

%trainedDetector = trainFastRCNNObjectDetector(trainingData,network,options)
trainingDataTable = objectDetectorTrainingData(gTruth3);
trainingData=trainingDataTable;


%cargar red
data = load('rcnnStopSigns.mat', 'stopSigns', 'fastRCNNLayers');
%stopSigns = data.stopSigns;
fastRCNNLayers = data.fastRCNNLayers;

%opciones de entrenamiento

options = trainingOptions('adam', ...
    'InitialLearnRate', 0.001, ...
    'MaxEpochs', 100, ...
    'VerboseFrequency', 10, ...
    'MiniBatchSize', 6, ...
    'Shuffle', 'every-epoch', ...
    'Plots', 'training-progress', ...
    'ExecutionEnvironment','gpu');

trainedDetector = trainFastRCNNObjectDetector(trainingData,fastRCNNLayers,options)
```

Ilustración 20 - Programación detector complejo

Este proceso de entrenamiento es más lento a pesar de que en este caso se ha utilizado la GPU, pero al ser más datos, es un proceso más lento el cual cambiando las opciones de frecuencia y el máximo de épocas puede cambiar, pero cuanto más tiempo de entrenamiento, el detector es más fiable.

Una vez obtenido el detector, se cambia a la zona de visualización en la cual se van a realizar las pruebas como se ha comentado anteriormente. La diferencia en este caso es el añadido de scores, esta información existe dentro de nuestro detector y en vez de tener una salida de etiquetado con el nombre del objeto como se observaba en el programa anterior, en este ejemplo la salida va a ser un recuadro sobre el objeto, pero en vez de aparecer el nombre de la etiqueta, va a aparecer el porcentaje de acierto del objeto, así se puede ver si el entrenamiento es o no es fiable, ya que depende del porcentaje que aparezca.

El proceso es el mismo, se lee una imagen guardada o se realiza una captura de pantalla de la cámara que se ha elegido. Tras tener la imagen a procesar, se va a incluir en la imagen el detector para reconocer tras el entrenamiento los objetos etiquetados, en el que dentro de bboxes viene la información de los recuadros y dentro de scores, viene la información de los porcentajes. Para terminar, se muestra la imagen procesada por el detector para poder observar la fiabilidad de este.

```
%% zona visualización

I = imread('Prueba8.JPG');

[bboxes,scores,label] = detect(trainedDetector,I);

if(~isempty(bboxes))
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
end
figure
imshow(I)
```

Ilustración 21 - Programación visualización

Como en el caso anterior, la última zona del programa es la zona de localización, ya que una vez se ha detectado el objeto, se geolocalice. Para ello se ha creado un apartado con una prueba de cómo se podría realizar. Al tener los datos de la altura a la que está el dron y la distancia a la que está el objeto (medidas con anterioridad al tomar las imágenes), se utiliza el teorema de Pitágoras para conseguir el dato de la hipotenusa ya que, uniendo la altura y la distancia al objeto, se forma un triángulo. Al resolver este teorema, se obtiene la distancia del objeto detectado y el dron.

```
37    %% zona ubicación
38    Vertical = 4;
39
40    Horizontal = 2,5;
41
42    Dist_drone = sqrt(Vertical^2+Horizontal^2)
43
```

Ilustración 22 - Programación localización

5. Resultados

En el momento que se ha terminado el proceso de etiquetado y entrenamiento de redes, es hora de enseñar los resultados obtenidos. En el procedimiento se ha indicado las redes finales que se han utilizado en la que se han obtenido los mejores resultados, sin embargo, se han hecho pruebas con más redes neuronales como la yolov2 y la resnet, de las que se comentarán los resultados a continuación.

La primera prueba que se realizó fue con la red yolov2, está red es la más usada dentro de las redes y la que más ha ido evolucionando a lo largo de los años por su alta velocidad de aprendizaje a pesar de perder fiabilidad en los resultados, ya que, al aumentar la velocidad, se pierden los porcentajes de acierto en los entrenamientos.

```
Training a YOLO v2 Object Detector for the following object classes:

* bottle

|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:06 |         7.20 |         51.9 |          0.0100 |
|       1 |          10 |       00:00:22 |         1.94 |          3.8 |          0.0100 |
|       1 |          20 |       00:00:40 |         1.81 |          3.3 |          0.0100 |
```

Ilustración 23 - Entrenamiento red yolov2 1º

En la imagen, están representadas las opciones de entrenamiento que se han elegido de forma global para todos los entrenamientos de las redes probadas. En la primera columna, se identifican las épocas que va a realizar el entrenamiento, cuantas más épocas, mayor es la duración del entrenamiento por lo que en casos de las redes rápidas, se ha escogido épocas de 10 para poder observar su comportamiento.

La segunda columna hace alusión a las iteraciones que realiza en cada época, las imágenes que entran y se entrenan dentro de la red neuronal, por ello la explicación de a mayor número de épocas, mayor número de iteraciones por lo que mayor es el tiempo de entrenamiento.

La tercera columna, se observa el tiempo que tarda cada iteración, el cual a medida que van avanzando, irá disminuyendo el tiempo entre iteración y aumentando el tiempo global del entrenamiento.

En las dos siguientes columnas (la cuarta y quinta columna) se presenta la información más relevante a la hora de los entrenamientos. Esto se debe a que en estas columnas se calcula el gradiente de entrenamiento según un mini conjunto de imágenes para su entrenamiento. La cuarta columna muestra el cómputo global del entrenamiento de la red, pero la columna más importante en este caso es el mini-batch los, ya que indica como de bueno está siendo el entrenamiento.

La opción ideal se daría al terminar el entrenamiento el número sea lo más cercano al 0 posible para comprender que el entrenamiento ha sido fructífero y ha sido descendiente en todo momento. Por último, la columna del entrenamiento base, la cual se establece un número base de entrenamiento, para saber la velocidad de entrenamiento. Este número se cambia una vez se hayan obtenido resultados y no son los esperados para cambiar la base del entrenamiento.

```
|       9 |         800 |       00:31:05 |         1.31 |          1.7 |          0.0100 |
|       9 |         810 |       00:31:22 |         1.44 |          2.1 |          0.0100 |
|      10 |         820 |       00:31:42 |         1.34 |          1.8 |          0.0100 |
|      10 |         830 |       00:31:58 |         1.24 |          1.5 |          0.0100 |
|      10 |         840 |       00:32:15 |         1.32 |          1.7 |          0.0100 |
|      10 |         850 |       00:32:31 |         1.35 |          1.8 |          0.0100 |
|      10 |         860 |       00:32:47 |         1.53 |          2.3 |          0.0100 |
|      10 |         870 |       00:33:05 |         1.15 |          1.3 |          0.0100 |
|      10 |         880 |       00:33:22 |         1.50 |          2.2 |          0.0100 |
|      10 |         890 |       00:33:38 |         1.48 |          2.2 |          0.0100 |
|      10 |         900 |       00:33:56 |         1.43 |          2.0 |          0.0100 |
|      10 |         910 |       00:34:09 |         1.60 |          2.6 |          0.0100 |
|=========================================================================================|
Training finished: Max epochs completed.
Detector training complete.
*********************************************************************
```

Ilustración 24 - Entrenamiento yolov2 2º

Cuando se termina el entrenamiento se observan los resultados obtenidos y con los que se van a contar en el detector. Como se ha comentado anteriormente, se ha entrenado una red rápida, por lo que se ha entrenado una base de datos de 1400 imágenes en 34 minutos. En cuanto tiempo, se cumplen los objetivos de la red, la parte del entrenamiento que se nota que no hay un buen entrenamiento es en el mini-batch loss, ya que como se ha explicado anteriormente, se está muy lejos del ejemplo ideal, habiendo picos de entrenamiento entre el 1.3 hasta el 2.6 por lo que acabando el entrenamiento en este último número. Por lo que, se podría concluir que no se van a tener buenos resultados con la yolov2.

Ilustración 25 - Resultado yolov2 interna a la red

La primera prueba que se realizó es con una imagen interna de la red, esto significa que la imagen es propia de la red entrenada, con el fin de observar si el entrenamiento es válido, en este caso, se nota que se ha recuadrado la botella con el porcentaje de aprendizaje de este. Al tener un resultado positivo con un ejemplo de dentro de la base de datos, se procede a hacer el mismo paso con una imagen externa a la red. Como se verá a continuación, el detector entrenado en esta red no conseguirá reconocer el objeto, concluyendo como se esperaba con los resultados obtenidos que esta red no es la óptima para la identificación y detección de botellas.

Ilustración 26 - Resultado yolov2 Externa a la red

Al observar que con la red Yolov2 no se obtenían los resultados esperados, se decidió utilizar otra red neuronal para comprobar el comportamiento de esta y conseguir unos resultados válidos. En el siguiente caso, se va a emplear la red resnet, se eligió por tener una estructura parecida a la de la Yolov2, pero en este caso es una red más lenta.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Mini-batch Loss	Base Learning Rate
1	1	00:00:12	17.83	317.9	0.0100
1	10	00:00:36	7.64	58.4	0.0100
1	20	00:01:00	9.12	83.1	0.0100
1	30	00:01:23	13.81	190.6	0.0100
1	40	00:01:46	10.74	115.4	0.0100
1	50	00:02:10	10.95	119.9	0.0100
1	60	00:02:34	11.75	138.0	0.0100
1	70	00:02:57	8.56	73.2	0.0100
1	80	00:03:21	9.02	81.4	0.0100
1	90	00:03:44	10.61	112.5	0.0100
2	100	00:04:10	13.66	186.7	0.0100
2	110	00:04:34	10.55	111.2	0.0100
2	120	00:04:57	12.34	152.2	0.0100

Ilustración 27 - Entrenamiento Resnet

Al comparar los resultados de la red resnet, con la de la red Yolov2, se puede notar que los tiempos son mayores, y se observa como el mini-batch Loss tiene picos de gran tamaño, desde los 58.4 hasta 317.9, sabiendo que es el inicio del entrenamiento, el entrenamiento debería ser descendente, pero al ver que oscilaba tanto y no se estabilizaba ni se acercaba al 0, se entiende que no va a mostrar resultados, se realizó la prueba con una imagen de dentro de la red entrenada como en el caso anterior.

Pero en esta situación, no se lograba reconocer una imagen que ya se ha entrenado, por lo que se desechó la red neuronal resnet.

Al obtener resultados no deseados en los dos entrenamientos de redes neuronales anteriores, se aposto por una red y un detector más simple, el cual no es necesario un entrenamiento tan grande.

```
ACF Object Detector Training
The training will take 4 stages. The model size is 18x42.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
--------------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 371 positive examples and 1855 negative examples...Completed.
The trained classifier has 256 weak learners.
--------------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 1855 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 371 positive examples and 1855 negative examples...Completed.
The trained classifier has 512 weak learners.
--------------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 1855 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 371 positive examples and 1855 negative examples...Completed.
The trained classifier has 1024 weak learners.
--------------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 1855 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 371 positive examples and 1855 negative examples...Completed.
The trained classifier has 2048 weak learners.
--------------------------------------------
ACF object detector training is completed. Elapsed time is 271.8592 seconds.
```

Ilustración 28 - Entrenamiento ACF Detector

En este caso, el entrenamiento es diferente, ya que las únicas variables para este tipo de entrenamientos son las imágenes etiquetadas y la segunda incógnita se puede elegir entre quitar los falsos positivos o escoger los casos positivos, los que carecen de errores. Por lo general, se suele escoger la primera opción, pero, al hacer varias pruebas con las diferentes opciones, se dieron mejor resultados con la elección de escoger los casos positivos.

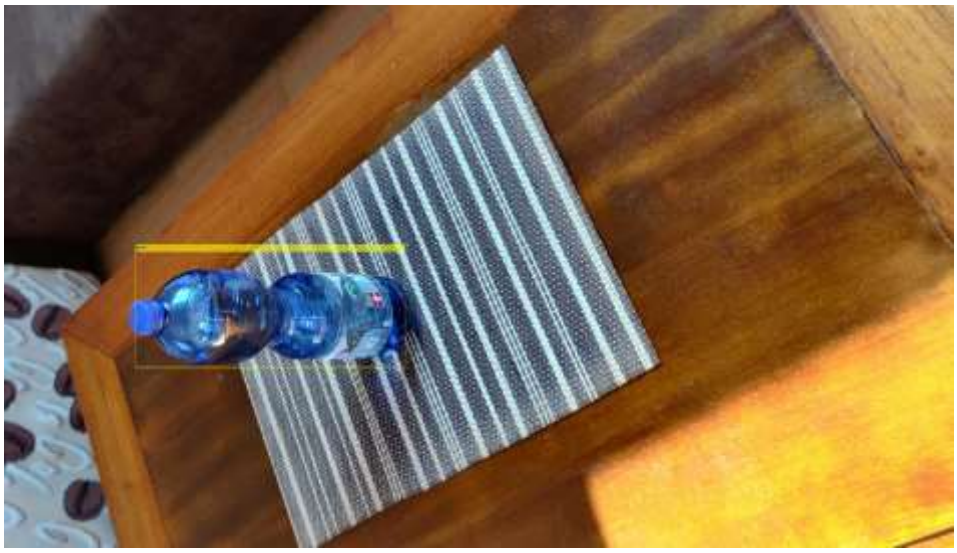

Ilustración 29 – Resultados ACF Detector interno

Para comprobar que el detector entrenado obtiene buenos resultados, se hizo la misma prueba que en los casos anteriores, se escogió una imagen interna a la red entrenada para ver cómo se comportaba, y se observa que los resultados son los esperados, consigue detectar los objetos internos a la red. Al conseguir esto, el proceso que se siguió es el mismo, probar que el detector funciona para imágenes externas de la red.

En los ejemplos externos de la red se van a utilizar imágenes a distintas alturas para ver el comportamiento de este, en este caso, se ha utilizado la imagen que está a 2,5 metros de distancias horizontal y a una altura de 0 metros, se ha recogido la imagen con el dron a la altura mínima posible. La altura va a ir cambiando en las imágenes ya que el dron va a tener que detectar los objetos a una distancia elevada. Por ello se harán pruebas desde los 0 metros hasta los 7 metros teniendo la distancia horizontal sin variar.

Ilustración 30 - Resultados ACF Detector 2 metros

Como se puede observar en la imagen, el detector funciona en imágenes externas a la red entrenada. En esta ocasión se ha escogido la imagen a 2 metros de altura ya que como se ha observado anteriormente, la imagen de prueba a 0 metros de altura no consigue tener resultados satisfactorios, sin conseguir detectar el objeto.

Ilustración 31 - Resultados ACF Detector 4 metros

Al tener resultados positivos en la imagen a 2 metros de altura, se hizo la prueba de seguir aumentando la altura, en esta ocasión, se hizo la prueba con una foto a 4 metros de altura, viendo que, a pesar de aumentar la distancia, el detector sigue funcionando, aunque también se puede observar que, al aumentar la altura, ya no detecta toda la botella, si no la parte más relevante de la misma, dejando sin detectar la tapa de la botella.

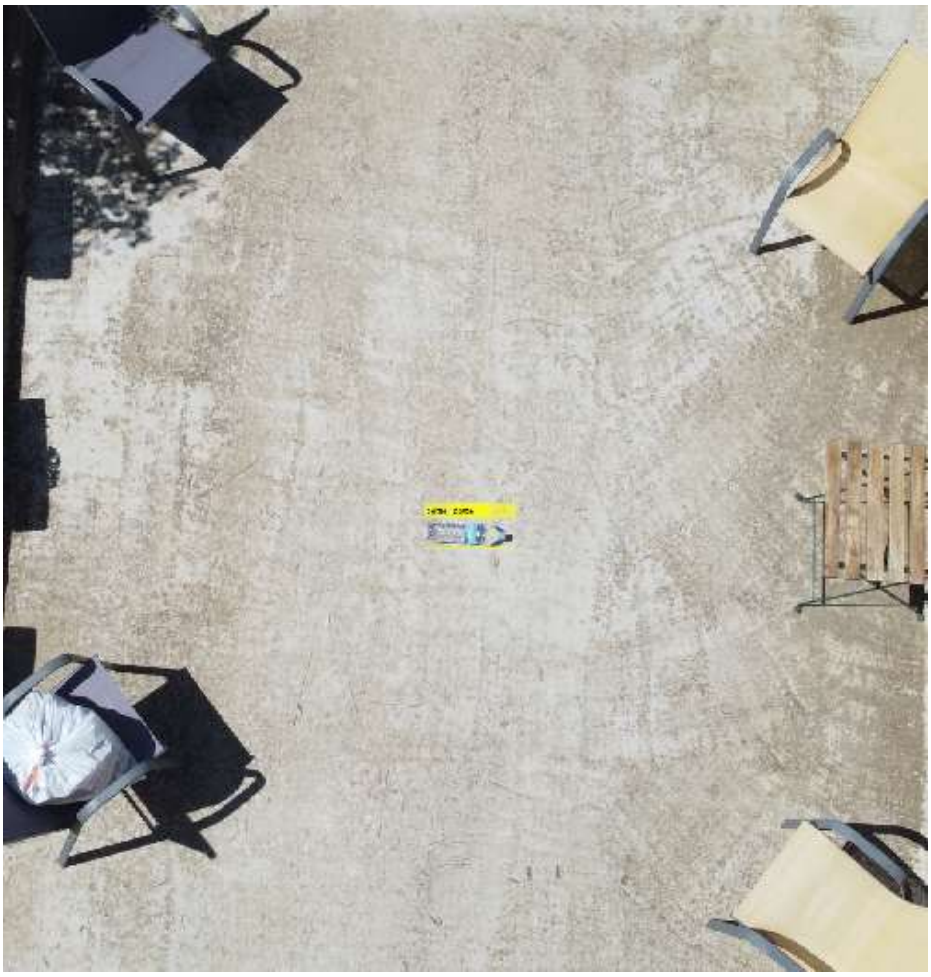

Ilustración 32 - Resultados ACF Detector 5 metros

Por un error que se generó al entrenar la red neuronal RCNN el cual se explicará más adelante, se hizo un segundo etiquetado de imágenes y se volvió a pasar por el entrenamiento, para ver si con el cambio en la base de datos, se podían mejorar los resultados. Pero, como podemos observar en la imagen, detecta el objeto dos veces, tiene dos etiquetas sobrepuestas. Esto supone que se retire el cambio en el recuadrado y restaurar el antiguo con el que se consiguieron resultados más limpios, ya que el detector funciona ya que detecta el objeto, pero al tener ese pequeño error, se decidió desestimar el nuevo etiquetado y mantener el antiguo.

Ilustración 33 – Resultados ACF Detector 7 metros

Por último, se hizo la prueba a 7 metros de altura, el detector funciona de manera correcta, pero, como se ha dicho anteriormente, se pierde detalle a la hora de realizar el etiquetado, además, al subir tanto de altura, no se puede apreciar lo que sale en la etiqueta, da igual si sale un porcentaje o una palabra. Esta casuística es algo a tener en cuenta a futuro, ya que, si se quieren detectar varios objetos diferentes, se tendrá que realizar por colores o realizando un recorte de la imagen y aplicándole zoom para poder reconocer que se está detectando.

Anteriormente se ha explicado el proceso de dos entrenamientos diferentes, el primero, ya se ha visualizado y otro que necesita un detector preentrenado y una red neuronal con el fin de obtener un detector más potente y fiable. En este caso, se va a utilizar la red RCNN, otro entrenamiento elegido por su rapidez y su fiabilidad al ser una estructura más estable.

A la hora de realizar el entrenamiento de esta red, aparecieron varios problemas. Ya que, al intentar entrenar, se obtenía un error y un warning, por falta de capacidad y de espacio en la GPU. Este error no se había localizado en las redes anteriores por lo que se investigó a que se debe la imposibilidad de realizar este entrenamiento. Se realizaron varios intentos cambiando tanto el tipo de entrenamiento (cambiar de entrenar con GPU a entrenar con CPU arriesgando la velocidad de aprendizaje de la red) y el propio código. Hasta que se revisó la estructura de la red neuronal RCNN y se descubrió que el error se debe a que las imágenes tenían distintas dimensiones. Al tener una base de datos grande y tomar las imágenes diferentes días con diferentes calidades, se tuvo que realizar el dimensionado de las imágenes para volver a realizar el etiquetado y su posterior entrenamiento.

100	25400	01:05:54	0.2590	92.97%	0.75	0.0010
100	25410	01:05:56	0.0319	99.22%	0.74	0.0010
100	25420	01:05:57	0.3840	89.84%	1.03	0.0010
100	25430	01:05:59	0.1501	95.31%	0.80	0.0010
100	25440	01:06:00	0.2886	92.19%	0.78	0.0010
100	25450	01:06:02	0.0455	99.22%	1.39	0.0010
100	25460	01:06:03	0.2380	97.66%	1.17	0.0010
100	25470	01:06:05	0.1740	96.09%	1.08	0.0010
100	25480	01:06:06	0.4335	85.94%	0.90	0.0010
100	25490	01:06:08	0.2092	93.75%	0.73	0.0010
100	25500	01:06:09	0.2243	93.75%	0.75	0.0010

Training finished: Max epochs completed.

Ilustración 34 - Resultados RCNN

Como se ha observado en ocasiones anteriores de entrenamiento de redes más complejos, se han propuesto unas opciones para reconocer el entrenamiento que se realiza. En esta ocasión, se ha añadido una nueva columna con el fin de conocer el porcentaje de aprendizaje de nuestra red, llegando a valores de 99%, aunque se ven mini. Se están utilizando redes rápidas, y en la tercera columna, se tienen los tiempos de esta red. Unos tiempos mayores a los demás entrenamientos, para poder ver la diferencia en el tiempo de entrenamientos en cuanto a resultados. Como se ha comentado anteriormente, la parte importante en los entrenamientos es la columna del mini-bacth los, en este caso, la penúltima columna, en la que se pueden observar valores inferiores a 1, teniendo picos más pequeños, por lo que es un entrenamiento a tener en cuenta.

Ilustración 35 - RCNN Detector 1 metro

En los resultados de las redes anteriores, se han utilizado imágenes internas a la red para comprobar si el entrenamiento de esta ha sido sólido y fiable, pero en esta situación al comprobar los resultados, y ver que el entrenamiento es sólido y con un detector fiable. Al comprobar el detector en las imágenes externas a la red, se observa que solo se detecta la imagen a 1 metro de altura. En la etiqueta del objeto, viene el porcentaje de acierto de la red, pero en este caso, sucede como en los resultados anteriores, al influir las alturas, influye en la detección, no se aprecia el porcentaje de la etiqueta y únicamente se detecta la mitad del objeto.

En este proyecto se han entrenado 4 redes neuronales distintas, para comprobar cuál es la que aporta mejores resultados. Tras ver el comportamiento de todas ellas, se ha decidido escoger la red neuronal más rápida y sencilla, ya que ha otorgado los mejores resultados, consiguiendo detectar 4 imágenes a diferentes alturas de 5 externas a la red se obtuvo como fallo la imagen a una altura de 0 metros. Las redes yolov2 y resnet, no nos otorgaron válidos, la primera no consiguió detectar ninguna de las imágenes externas a la red, mientras que la segunda, no fue capaz ni de detectar imágenes internas a la red. Por último, la red RCNN en la cual se cambió la base de datos de las imágenes y solo se obtuvo el resultado de 1 imagen de 5 externas a la red por lo que se decidió explicar el procedimiento de las dos redes de las que se obtuvieron resultados.

6. Estudio de Viabilidad

Durante el transcurso del proyecto, se han mencionado varios utensilios para el desarrollo de este, en especial, los mencionados en el apartado 3. Por lo que, al ser un proyecto de investigación los costes asociados al proyecto para saber la inversión del proyecto son los relacionados con las licencias empleadas y con los costes horarios.

Gasto	Detalle	Importe
Remuneración Ingeniero Junior		7.000 €
	200 horas a 35€/h	
Coste licencia software		3.500 €
	Deep Learning toolbox	-
	Computer Vision toolbox	-
	Image processing toolbox	-
	Statistic and machine learning toolbox	-
	GPU Coder	-
Coste hardware		1.089 €
Coste licencia dron		2.000 €
Coste dron		1.500 €
Permiso vuelo		500 €
	TOTAL	**15.589 €**

Ilustración 36 - Tabla Importe Total

Como se puede observar en la tabla, los primeros costes del proyecto a los que se hacen mención son los costes del trabajador, las horas totales invertidas en desarrollar el programa y el procedimiento, en este caso, la duración de la programación es de dos meses, alrededor de 200 horas multiplicado por el valor por hora de un Ingeniero especializado en este software. En España, el sueldo por hora para este puesto es de 35 € la hora, por lo que, multiplicando las horas por el sueldo a la hora, se saca un coste de 7.000€.

Los costos de la licencia del software, como se ha mencionado a lo largo del proyecto, la aplicación utilizada para el desarrollo del proyecto es Matlab. En este caso, se ha utilizado la versión estudiante/académica, una versión utilizada para la investigación dentro de los centros por lo que el coste es menor. Al ser una de las finalidades de este proyecto enseñar a utilizar las redes neuronales en Matlab, se han incluido los costes para su uso comercial o de una organización externa a la educación. En este caso, se vislumbraban dos opciones. La primera opción incluía en la licencia todos los productos que otorga MathWorks, incluyendo Matlab y Simulink, lo cual tiene un coste anual de 3.500€ la licencia individual, una licencia pensada para startups. Y la segunda opción, al conocer las toolbox necesarias para la realización del proyecto, se ha creado un presupuesto únicamente incluyendo los productos necesarios para la realización de este proyecto lo que supone un coste de 2.300€.

En esta ocasión se ha escogido la primera opción ya que, al ser una licencia anual, se va a poder aprovechar más la licencia completa. Ya que, al terminar el proyecto, se puede seguir utilizando el software con distintas toolbox en caso de querer empezar un proyecto diferente. En cambio, al tener únicamente las partes de la licencia necesarias para este proyecto, al terminarlo, se cerrarían las puertas a proyectos diferentes. Por lo que se ha decidido que, a pesar de tener un precio elevado, a futuro, comprar la licencia completa es más rentable.

Una vez se ha explicado los costes del software, se ha de estimar los costes del hardware que se ha utilizado. Estos costes incluyen el propio ordenador sobremesa, contando la CPU y la tarjeta gráfica y los periféricos, referidos al monitor, el teclado y el ratón. El importe del hardware equivale a 1.089€ en el momento de su compra. A futuro comprar estos elementos será más asequible o por el mismo precio tener un ordenador más potente.

En este proyecto se ha utilizado un dron para la toma de imágenes y con el fin de utilizar el programa para automatizar la detección de objetos en tiempo real. Como se ha comentado anteriormente, este proyecto se ha realizado con la empresa IBTICAE por lo que se ha usado un dron fabricado por ellos valorado en 1.500€. Una vez se tiene el importe del dron, es necesario contar con licencia de vuelo, y un permiso de vuelo. Esto es necesario debido a la normativa de drones en España ya que se ha empezado a aplicar los Reglamentos Europeos y si el dron pesa más de 250 gramos es necesario tanto el permiso de vuelo como la licencia de piloto de dron. Esto influye en los costes del proyecto, aumentando el importe por la licencia 2.000€ y por el permiso de vuelo 500€.

El coste final de este proyecto de investigación sobre la detección de objetos utilizando un dron, se ha explicado a lo largo de este punto obteniendo varios valores, si se suman dichos resultados, se obtiene un importe total de 15.589€.

7. Geolocalización

La última parte de este proyecto es la investigación de la geolocalización de los objetos los cuales se han detectado anteriormente y conocer su posición exacta en el mapa, con sus coordenadas.

Esta parte es la más complicada, ya que en el software que se ha utilizado no hay ninguna aplicación ni ningún comando especifico, y no constan estudios sobre lo mismo ni investigación por lo que este apartado se quedó como punto de mejora y de investigación.

Para poder lograr la localización de un objeto en 3 dimensiones a través de una foto cuya salida es en 2 dimensiones, se han investigado varias formas de realizar la geolocalización tanto utilizando el software o utilizando otros métodos.

El primer método que se investigó es a través del Teorema de Pitágoras. Para ello se pondría la cámara en la esquina superior izquierda, el objeto se situará en el final de la hipotenusa, y el ángulo de 90º se formará con el suelo. El drone cuenta con un sistema de GPS el cual se facilita su ubicación exacta, con ese dato y con la implementación de un sensor de distancias, apuntando hacia el final de la hipotenusa, en donde se encontraría el objeto, se tendría la posición exacta del objeto. Para ello, se usaría el dato de la distancia dado por el sensor como si fuese la hipotenusa y el cateto opuesto (el cateto vertical), viene dado con la altura a la que está el dron, ya que el objeto estará situado a la altura del suelo. Por lo que aplicando el Teorema de Pitágoras se sacaría el dato del cateto contiguo (el cateto horizontal). Una vez se han obtenido estos dos datos, se sumaría el dato calculado del cateto contiguo a la ubicación del drone y se restaría el dato del cateto contiguo a la ubicación, y se obtendría la ubicación del objeto detectado [31].

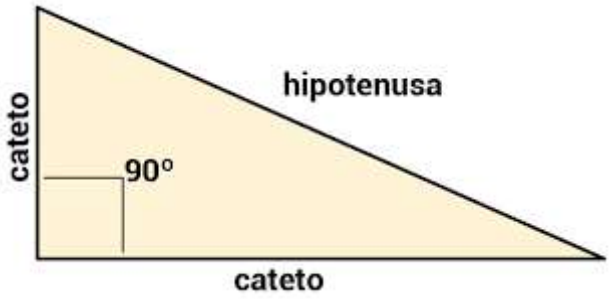

Ilustración 37 - Teorema de Pitágoras

En este caso, también se consideró la opción de mantener el dron justo encima del objeto detectado, y la cámara apuntando directamente hacia abajo en perpendicular al dron. Con esta posición, se puede obtener la información de la ubicación del objeto con respecto del dron ya que únicamente habría que restar la altura a la que se encuentra el dron y se obtendría la ubicación del objeto gracias a conocer la posición del dron a través del GPS.

La problemática que se encontró con este modelo es que la cámara no se va a estar quieta en una misma posición ya que con el movimiento del drone se puede mover o incluso si se utiliza una cámara que se mueve 180º por ejemplo, este sistema se rechazaría, por lo que solo se podría utilizar para un entorno ideal.

Otra manera de realizar la ubicación de los objetos es la conversión de datos. Las imágenes se miden en pixeles mientras que en un mapa se miden en coordenadas. Este suceso genera que el objeto que se detecta a través de las redes neuronales interno en la imagen aporta su ubicación en píxeles. Pero se necesita su ubicación en coordenadas, para ello, al tener dos variables diferentes, se va a tener que aplicar un proceso de conversión mediante la combinación de metadatos geoespaciales con imágenes.

Para poder conseguir esta conversión de datos, se necesita registrar una zona de vuelo en la que el drone se mueva siguiendo una ruta, para que el GPS registre la información del vuelo y poder comenzar con la conversión de los píxeles a coordenadas de latitud y longitud. Es importante que se mencione que la recogida de la ruta, el dron tiene que estar a 90º con el suelo en todo momento, y la cámara mirando hacia abajo cumpliendo el ángulo mencionado anteriormente para que se pueda obtener una información más completa y fiable [17].

Para el proceso de conversión son necesarias tres variables que se van a extraer del registro del vuelo, estas son: las coordenadas del drone, la altitud a la que está en todo momento el drone y la ruta que sigue el drone con su brújula para saber la orientación de este. El último dato a conocer es el giro de la cámara, en este caso al ser una parte de investigación con la empresa IBTICAE, la cual cuenta con un dron cuya cámara llega a girar 60º se va a utilizar ese dato [17].

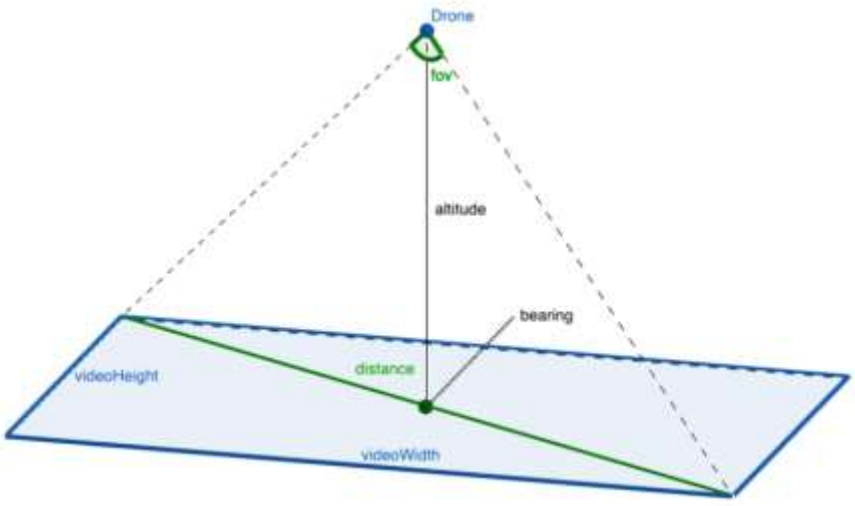

Ilustración 38 - Conversión de pixeles a coordinadas

A través de los registros del vuelo se puede obtener los datos de la altitud, es una variable que, a expensas de tener un GPS, se puede obtener a través de la señal de un sensor, midiendo continuamente el terreno por el que pasa el dron. La variable FOV, es el punto de vista de la cámara que se va a utilizar y el rango de esta. Si se multiplica la altitud medida por la tangente del ángulo de la cámara, se obtiene la distancia existente entre las dos esquinas de la imagen que se va a utilizar para la conversión. El último cálculo necesario es conseguir la relación de aspecto, se refiere a la relación entre la altura y el ancho de la imagen medida, por lo que se va a realizar el arco tangente de la división entre la altura y la anchura [17].

Una vez se llega a este punto, se necesita utilizar el método destino para calcular todos los puntos del GPS, desde las esquinas para que se pueda utilizar la mayor información posible y se utilice la mayor fiabilidad posible. La medida siempre se va a comenzar desde el centro, justo debajo del dron como se observa en la imagen, y lo que se va a realizar es recorrer la recta distancia desde el punto central hasta una de las esquinas utilizando un direccionamiento para que se pueda reconocer en qué punto se encuentra el objeto y el reconocimiento. Con este paso, se puede conseguir convertir cualquier punto de la imagen en un punto del GPS, utilizando los cálculos de la distancia y el ángulo al que va a estar la cámara orientada [17].

En el caso de este proyecto, se propone tener la cámara quieta, pero al igual que se comentaba en el caso anterior, siendo un dron, es muy costoso el conseguir que mantenga un rumbo estable, por lo que la mayoría de las cámaras tienen varios ángulos de inclinación. Si se consiguiera esa estabilidad esta opción es la más fiable, pero si la cámara va a estar en constante movimiento, al programa habría que añadirle una opción de indicar el ángulo al que está la cámara en todo momento y tome todos los detalles en tiempo real para que al poder detectar y geolocalizar el objeto se consiga el menor error posible.

8. Conclusiones

La realización del presente proyecto ha desembocado en una serie de conclusiones que podrían ser un buen punto de partida para futuros desarrollos. A continuación, pasamos a describir algunas de las mismas:

- Se ha creado una aplicación basada en un entorno de programación abierto para la automatización del proceso de detección de objetos en imágenes mediante algoritmos de visión artificial.

- A la hora de realizar el etiquetado de imágenes, cada red neuronal tiene un comportamiento diferente, siendo unas redes más tolerantes que otras. Al encontrar el error de las imágenes se determinó que antes de comenzar cualquier tipo de manipulación de imágenes antes hay que redimensionarlas para que no vuelva a aparecer este tipo de errores/ warnings en la programación.

- Para este trabajo se han utilizado una Base de Datos de 1400 imágenes, un número de fotos suficiente para poder entrenar cualquier red neuronal sin ningún problema, pero hay que tener en cuenta que cuantas más imágenes se utilicen mayor estabilidad y solidez tendrá la estructura que se entrene y más fiabilidad a la hora de obtener resultados.

- Tras la realización de pruebas con distintas redes, según su estructura se van a tener diferentes resultados. Los resultados más notables se han dado utilizando el entrenamiento más simple, el ACF detector. El cual se ha impuesto consiguiendo detectar imágenes tanto internas como externas a la red con un 83% de acierto. La otra red con la que se han obtenido resultados es la RCNN, la cual ha obtenido un 17% de acierto, por lo que la red más simple ha sido la que mejores resultados ha obtenido.

- Al utilizar un dron para la toma de imágenes, las alturas y distancias van a variar, por lo que es una variable a tener en cuenta a la hora de la geolocalización. No se va a tener un sistema ideal por lo que se va a tener que utilizar el método de conversión de datos, transformando los píxeles a coordenada para poder llevar a cabo la localización de los objetos detectados.

Para concluir, hay que mencionar que la investigación iniciada en este proyecto resulta muy prometedora y útil para labores medioambientales faltaría seguir con la parte de implementación en el hardware enmarcado en el dron para conseguir la detección en tiempo real.

9. Bibliografía

[1] “El futuro de la tecnología: drones y la inteligencia artificial - 2023". DRONES ESPAÑA. https://xn--drones-espaa-khb.eu/el-futuro-de-la-tecnologia-drones-y-la-inteligencia-artificial/

[2] corjuelag. "Beneficios de la creación de drones con inteligencia artificial." Portal de noticias de tecnología, Realidad Virtual, Aumentada y Mixta, Videojuegos. https://niixer.com/index.php/2023/04/11/beneficios-de-la-creacion-de-drones-con-inteligencia-artificial/

[3] "Este dron acuático recoge una tonelada de plástico al día gracias a la IA". Escudo Digital. https://www.escudodigital.com/tecnologia/inteligencia-artificial/este-dron-acuatico-recoge-tonelada-plastico-dia-gracias-ia_51802_102.html

[4] Colaboradores de los proyectos Wikimedia. "Historia de la aviación - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Historia_de_la_aviación#Siglo_XVIII-siglo_XIX:_Aeronaves_más_ligeras_que_el_aire

[5] "Complete Evolution & History of Drones: From 1800s to 2023". Propel RC. https://www.propelrc.com/history-of-drones/#History_of_Drones_The_First_Drone

[6] Cienciorama. http://www.cienciorama.unam.mx/a/pdf/535_cienciorama.pdf

[7] "A brief history of drones: from pilotless balloons to roaming killers". Interesting Engineering | Technology, Science, Innovation News and Videos. https://interestingengineering.com/innovation/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs

[8] Colaboradores de los proyectos Wikimedia. "Visión artificial - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Visión_artificial

[9] "Redes neuronales profundas preentrenadas- MATLAB & Simulink- MathWorks España". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink – MATLAB & Simulink. https://es.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[10] "Object Detection Using YOLO v4 Deep Learning- MATLAB & Simulink- MathWorks España". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/help/vision/ug/object-detection-using-yolov4-deep-learning.html

[11] "NVIDIA Jetson Nano para aplicaciones y materiales educativos basados en IA perimetral". NVIDIA. https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-nano/

[12] "Todo lo que necesitas saber sobre Jetson Nano de NVIDIA | BricoGeek.com". Blog BricoGeek.com. https://blog.bricogeek.com/noticias/robotica/todo-lo-que-necesitas-saber-sobre-jetson-nano-de-nvidia/

[13] "Red neuronal multicapa para la evaluación de competencias laborales". SciELO - Scientific Electronic Library Online. http://scielo.sld.cu/scielo.php?pid=S2227-18992016000500016&script=sci_arttext&tlng=pt

[14] "Aplicaciones de las redes neuronales. El caso de la Bibliometría | Ciencias de la Información". Ciencias de la Información. http://cinfo.idict.cu/index.php/cinfo/article/view/54

[15] ELP Best USB Webcam Module. http://www.webcamerausb.com/elp-8mp-3264x2448-sony-imx179-cctv-usb-camera-550mm-varifocal-cs-lens-hd-usb-industrial-box-inside-surveillance-usb-camera-webcam-p-240.html

[16] "Drones, su evolución a través de la historia - Bestechnology Group". Bestechnology Group. https://bestech-group.com/drones-su-evolucion-a-traves-de-la-historia/

[17] B. Dwyer. "Using Computer Vision with Drones for Georeferencing". Roboflow Blog. https://blog.roboflow.com/georeferencing-drone-videos/

[18] "Cómo medir distancias en 3D con Pix4D". Cursos de Teledetección, Drones y LIDAR - Formación presencial y online en Drones, teledetección y LIDAR. https://www.cursosteledeteccion.com/como-medir-distancias-3d-con-pix4d/

[19] "Medir la distancia entre píxeles en la app Image Viewer- MATLAB & Simulink- MathWorks América Latina". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://la.mathworks.com/help/images/measure-distance-between-pixels-in-image-viewer-app.html

[20] "Escala y mediciones sobre fotografías - Jose Pereira". Gestion de Color, fotografía científica, imagen digital - Jose Pereira. http://www.jpereira.net/gestion-de-color-articulos/mediciones-y-asignacion-de-escala-en-fotografias

[21] "3. Distancia de enfoque y tamaño de la imagen - foto igual". foto igual. http://fotoigual.com/3-distancia-enfoque-tamano-la-imagen/

[22] "? MIDIENDO la DISTANCIA entre 2 objetos | Python - OpenCV » omes-va.com". OMES. https://omes-va.com/midiendo-la-distancia-entre-2-objetos-python-opencv/

[23] "Jetson-Nano Search and Rescue AI UAV". Hackster.io. https://www.hackster.io/jonmendenhall/jetson-nano-search-and-rescue-ai-uav-9ca547

[24] marcelo pardo. cómo medir distancias desde una imagen. Método en 2D y 3D. Disponible: https://www.youtube.com/watch?v=4i8xjP89R7g

[25] Sistema de Información Científica Redalyc, Red de Revistas Científicas. https://www.redalyc.org/pdf/461/46154070004.pdf

[26] "Desarrollo de una metodología sencilla para la georreferenciación y medición de distancias a partir de imágenes de satélite sistemáticamente georreferenciadas". Welcome to AquaDocs. https://aquadocs.org/handle/1834/3743

[27] "Registration of video to geo-referenced imagery". IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/711963

[28] "ESurf - Short Communication: TopoToolbox 2 â€“ MATLAB-based software for topographic analysis and modeling in Earth surface sciences". ESurf - Recent. https://esurf.copernicus.org/articles/2/1/2014/

[29] "MATLAB Machine Learning". Google Books. https://books.google.es/books?hl=es&lr=&id=3kXODQAAQBAJ&oi=fnd&pg=PR6&dq=machine+learning+matlab&ots=ZMOzVOa9mP&sig=tOwzypHWF69qDaVWZjwAAZdiImY#v=onepage&q=machine%20learning%20matlab&f=false

[30] "Which is Better for Deep Learning: Python or MATLAB? Answering Comparative Questions in Natural Language". ACL Anthology. https://aclanthology.org/2021.eacl-demos.36/

[31] Lic. Diana Giménez de von Lücken, MSc. "Triángulo rectángulo y Teorema de Pitágoras (1) - Escolar - ABC Color". Noticias de Paraguay y el mundo de último momento hoy en ABC Color. https://www.abc.com.py/edicion-impresa/suplementos/escolar/triangulo-rectangulo-y-teorema-de-pitagoras-1-1788042.html

[32] "MATLAB - El lenguaje del cálculo técnico". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab